CERTIK

Security Assessment

Starlay

Apr 13th, 2022

Table of Contents

Summary

This report has been prepared for Starlay to discover issues and vulnerabilities in the source code of the Starlay project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Starlay
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/starlay-finance/starlay-stake/commit/1bdcd6bccf65555094a5a88c9e73084887cf7bf7 https://github.com/starlay-finance/starlay-stake/commit/b1b52389bba156211b1bf00f01a2208a606b682b https://github.com/starlay-finance/starlay-incentives-controller/commit/f93a9840eb4a3b5a6947ffd200b14b56da484150 https://github.com/starlay-finance/starlay-incentives-controller/commit/8417eb39181023f3b87c7b8cede6e71c2e18d9c0 https://github.com/starlay-finance/starlay-protocol/commit/2d5c8dfe8df1474ecb0c1f85e753c2f9c4a9e363 https://github.com/starlay-finance/starlay-protocol/commit/13b49b3fb458308de358ce5058a65ebe7b542e2a
Commit	

Audit Summary

Delivery Date	Apr 13, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0	0
● Major	4	0	0	3	0	0	1
● Medium	1	0	0	1	0	0	0
● Minor	7	0	0	4	0	0	3
● Informational	6	0	0	1	0	0	5
● Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
BIC	incentives-controller/contracts/incentives/base/BaseIncentivesController.sol	b17e0ed1f7b906ec28d0e82c5677fc3871e141dcbf5223d83224dd38a30d9dff
DMC	incentives-controller/contracts/incentives/base/DistributionManager.sol	e7295c289380557cae57c846a5deabda5b51c58f1bd294c9c9cd7c7c79ecbfcb
PRI	incentives-controller/contracts/incentives/PullRewardsIncentivesController.sol	c6845ecf4dcc34992a067196234a7d2cf6d08be36eebbb0d5b8d2a60ef6c5d45
STI	incentives-controller/contracts/incentives/StakedTokenIncentivesController.sol	a9992c7e50d8741f36b29f0de1fbd2355a8ddd0b61dc121d17f8aa141d3abf39
IDC	incentives-controller/contracts/interfaces/IDistributionManager.sol	1eea65ba64a950b7e7538a4a77fdd07334af092c2228e92ab137529bcfb63907
IGD	incentives-controller/contracts/interfaces/IGovernancePowerDelegationToken.sol	2946235903c46a9a8e9fc520eed9b1d3cf02f5a5807c32ffbd266e7f58843a9b
IIS	incentives-controller/contracts/interfaces/IIncentivesController.sol	0a83e98f2e4b6aff03e4654017776460b33d3968f21895ad00476b27232c9308
IIE	incentives-controller/contracts/interfaces/IIncentivesExecutor.sol	1aed0cb63089c289a9dfaf4f3e1e3635803586e48d0c66601f91a161c3626942
ILD	incentives-controller/contracts/interfaces/ILTokenDetailed.sol	67cd18112bcd8fe6777da53ebc8409ebf8fc35e4caf5ded0b7cc092ece40e54f
ILK	incentives-controller/contracts/interfaces/ILendingPool.sol	e73b86f461133cbac09e556a53b0c1aaf4d17159f27a39c99ee0ea5f9705f490
IPP	incentives-controller/contracts/interfaces/ILendingPoolAddressesProvider.sol	28c1ad89cd37055dd9aedfafa80db699af83f4893fae374ab8b0d30159ef9853
IPR	incentives-controller/contracts/interfaces/ILendingPoolAddressesProviderRegistry.sol	d65f779ffb15f5f40c7411ba8c73a785203fd9d93bf528d0968dca687f4f09b6
IPC	incentives-controller/contracts/interfaces/ILendingPoolConfigurator.sol	da9745ee697675df4ff3ca8269c70058d116a1bf7134b01e25761d04cbfca057
IPD	incentives-controller/contracts/interfaces/ILendingPoolData.sol	0986022c23a7d6888919c7caa7f382ffeb3e408150dd2b2c483b4c6d2b6cf389

ID	File	SHA256 Checksum
IPI	incentives-controller/contracts/interfaces/IProposalIncentivesExecutor.sol	43080fca64249cdbcd4ca62cc310ba7b254c01bf026296fa8831485d4a47cf13
ISS	incentives-controller/contracts/interfaces/IScaledBalanceToken.sol	6dd146f1c5072cd600b2e79d5631fb3bb5598c568b486889d5339c1ce2aa7746
ISW	incentives-controller/contracts/interfaces/IStakedTokenWithConfig.sol	748c28b7d1feaad3a52b02e8e80b720834d8bc5ee1e1508bfbf049e02181a82c
ISV	incentives-controller/contracts/interfaces/IStarlayGovernanceV2.sol	28e65c572ed785658286cb675272fe1bfc07821682da8fed42133a71befa0645
ISR	incentives-controller/contracts/interfaces/IStarlayRewardsVault.sol	22fdab34e67a407842e6056f0b023b922c3d47a41f5349f050e75ef195eff33c
LTC	incentives-controller/contracts/lending-pool/LToken.sol	d1c19077c8f474ef4e1a9edd9548b27fae79c8b1660a49c92807d3e37dd14968
LCS	incentives-controller/contracts/lending-pool/LendingPoolConfigurator.sol	44e47f7315a129e7951f7360ea8a2a04bfa3f5436455ce02e16d6e2b76bb370c
SPP	incentives-controller/contracts/lending-pool/StarlayProtocolDataProvider.sol	10459514c2c5e3fbc8c2770609faa08af35fa3b3ed811f6f5155bfe4feeb785a
VDS	incentives-controller/contracts/lending-pool/VariableDebtToken.sol	375da69d0565f8a422e9ad417e30c91236b0a49ddf1cd5832c74ab064ea5ead9
IRK	incentives-controller/contracts/stake-v1/contracts/interfaces/IERC20.sol	42c346056d38377aa31f6ff514b03c5afdf3ea0a6c71390f7fbfdde8e36d4b0d
ITS	incentives-controller/contracts/stake-v1/contracts/interfaces/ILToken.sol	eae38ce30b6ca2f0675c34ae5eab2be65d60dbc71af9dc21eeb90e6ebbf69f57
ISK	incentives-controller/contracts/stake-v1/contracts/interfaces/IStakedToken.sol	73a7b01257fc98922798794516684 86f7f55f231d1301e48463aa8f44f1bb188
VIP	incentives-controller/contracts/stake-v1/contracts/utils/VersionedInitializable.sol	f7cd8f414fb49e10e52ad37ee89d31c031c54144d6b644224a7100ab8df1e118
SLK	incentives-controller/contracts/stake/StakedLayV2.sol	e615ca0c6529fb9af9d80446980442e22549cd0df0c270d1d607ac3a69432e25
CSP	incentives-controller/contracts/utils/Context.sol	15ab2917843a31da12fdb311715ca93d5a191076c3ecd88afa7861fdfda51068

ID	File	SHA256 Checksum
DTC	incentives-controller/contracts/utils/DataTypes.sol	2a614322b7de123b2f7346dea2011f10ec021f3e1ef3ce90fa97709e66e942f4
IAP	incentives-controller/contracts/utils/InitializableAdminUpgradeabilityProxy.sol	69f4e298eb8cb47713652d456adba83a39b31fe4677909531e288d5487518d8a
MEC	incentives-controller/contracts/utils/MintableErc20.sol	560802b24a414345748858d0f6e4f3ed64ec2d62300fc1360d84a8c88f5011ff
PMC	incentives-controller/contracts/utils/PercentageMath.sol	e5be6c8d4d904733178a6548da14d9007191aee48799210d9b5f4595b76e1945
STP	incentives-controller/contracts/utils/SelfdestructTransfer.sol	3a9482506ef01107e7a0a5307a9cf315de47af042fd4350964ae2cea9fc8ce21
VIK	incentives-controller/contracts/utils/VersionedInitializable.sol	f7cd8f414fb49e10e52ad37ee89d31c031c54144d6b644224a7100ab8df1e118
ASK	protocol/contracts/dependencies/openzeppelin/contracts/Address.sol	084a79c320b8082a68a8d169937ea669aeeb1fe2d2c400ed9630f074b9201d4f
CSK	protocol/contracts/dependencies/openzeppelin/contracts/Context.sol	3505c57fbb7d5c1ed078bae3eae6e620435bcedab3601f6e2f4787e1d5c3db4b
ERS	protocol/contracts/dependencies/openzeppelin/contracts/ERC20.sol	c5fa87d34412474c79296156846eb7c75d2fafecd55feb524882c124415a8a26
IRS	protocol/contracts/dependencies/openzeppelin/contracts/IERC20.sol	cbf23bb29da4ab66a8b0f3a38f3a2c08f6a74ed13a654ed3943ea418a32a4897
IED	protocol/contracts/dependencies/openzeppelin/contracts/IERC20Detailed.sol	25f312fccf7eccd9f9302a1bd3155be4c6072cee3c4e8ef94c8fb6aba90d1ebe
OSC	protocol/contracts/dependencies/openzeppelin/contracts/Ownable.sol	cb04add9cbfcebde26cd4d0060a9e9fd933fc7839081c78f5dc96aeb1fdec8e6
RGS	protocol/contracts/dependencies/openzeppelin/contracts/ReentrancyGuard.sol	3fc7968f4a1937caf3c96dffbac350398f86faad96288502e02c3a2b9f245e39
SER	protocol/contracts/dependencies/openzeppelin/contracts/SafeERC20.sol	b187fa735ed62d5a4078dc4247704d811cec2b130987d65273dff648d493c8e0
SMC	protocol/contracts/dependencies/openzeppelin/contracts/SafeMath.sol	c3c781565b60a273d99a750e416a3790736a83281ac4b30c358dee6d7f31a426

ID	File	SHA256 Checksum
AUP	protocol/contracts/dependencies/openzeppelin/upgradeability/AdminUpgradeabilityProxy.sol	112daacac30e892fe8bb15a6e5ba42bbfc74f654b6de3c570fba7603b25b358b
BAU	protocol/contracts/dependencies/openzeppelin/upgradeability/BaseAdminUpgradeabilityProxy.sol	0b7356d4c5e6f54ca227ab9311156009e60ea28c210638378754e47f54422de5
BUP	protocol/contracts/dependencies/openzeppelin/upgradeability/BaseUpgradeabilityProxy.sol	636dfe18ddf3377a50e2c97d2e26df0db83a835a006564c0a6b1c431e1799f6e
ISC	protocol/contracts/dependencies/openzeppelin/upgradeability/Initializable.sol	a67a76516b36da7b8b2b350155d21c687ccca3b518960e4c183cd964559eec63
IAU	protocol/contracts/dependencies/openzeppelin/upgradeability/InitializableAdminUpgradeabilityProxy.sol	9cd9b08540965cdcbd3a96616c6742b8d046cea6bfb6b19f702e2ac60e02b471
IUS	protocol/contracts/dependencies/openzeppelin/upgradeability/InitializableUpgradeabilityProxy.sol	f6ce7f91f3194deac901457fb31aef5777b35d7947370c55064eaaa5e785ba7b
PSC	protocol/contracts/dependencies/openzeppelin/upgradeability/Proxy.sol	280985357d8577a76ccb10fe7cff56ae20189f2d14310c61f73058e0c1a7ed9c
UPS	protocol/contracts/dependencies/openzeppelin/upgradeability/UpgradeabilityProxy.sol	76557743694a0a7fa2f6738021734171156c634e8adb1019cb60d64dfe4cb4be
LTA	protocol/contracts/deployments/LTokensAndRatesHelper.sol	f5c666a5235edbdc120f4b6b12e83c258427425e13954bd6f200a7d0e286fa86
SAV	protocol/contracts/deployments/StableAndVariableTokensHelper.sol	52a504f7a97375ea01a961ab96fbc884fbc90b8843eb0b31a31df9abb7bf3290
SLC	protocol/contracts/deployments/StringLib.sol	ecc4df8b2cf6cf2ca6ec1ac53004a2c6e407d4ad8c2ebe4e5c179e905e5d140b
FLR	protocol/contracts/flashloan/base/FlashLoanReceiverBase.sol	c28c29813496c43aa086ef79e4082f72faa9e0a7710935a29680c4e115db6b3c
IFL	protocol/contracts/flashloan/interfaces/IFlashLoanReceiver.sol	934bc87ae2004f0f71ab9c9a307b7a0686494a2ea93aa5cd3c71b5c0deff3fc3
ICA	protocol/contracts/interfaces/IChainlinkAggregator.sol	1a4cd55fdd50b13ca6ed1643607fc6bf230bde4ca301dfb46a86766ae3947b89
ICD	protocol/contracts/interfaces/ICreditDelegationToken.sol	68b074f5751c42359fb81c6efc08ca04445ee9d80623572b3418eb82858f0fc8

ID	File	SHA256 Checksum
IDT	protocol/contracts/interfaces/IDelegationToken.sol	cf497b0991626bc5ad916227d506e2cbdfebd33f0ce0a1f126a37c2bd7cdf543
IDS	protocol/contracts/interfaces/IDiaAggregator.sol	62bfb9a2edf10d49d56e2c0a931cac53b19e97805bb45e8be10f103b0bcf4859
IEK	protocol/contracts/interfaces/IERC20.sol	4f5e9403be77447aa5c2d09814c6807bd75d590265d26719f797d5d6aac45775
IEW	protocol/contracts/interfaces/IERC20WithNonce.sol	b85e16ec87c9e2f3c7431b57e00dedb61240ca86dcd8e72d4895be8532144e88
IEP	protocol/contracts/interfaces/IERC20WithPermit.sol	2490e6769278c88eea483b093d1c72891dc223060b74fd51ee605173b2e18547
IID	protocol/contracts/interfaces/IInitializableDebtToken.sol	9a9ead038ec3679e21e278851ff1547a6d254690c740a181c70ed3759781e80e
IIL	protocol/contracts/interfaces/IInitializableLToken.sol	b18521c12ffee3a5a7129573786050581fa8538bdef1cbb2e71f26d311b7c923
ILT	protocol/contracts/interfaces/ILToken.sol	69669af5e4d99be265d0564c4dc58c6a9446961607c7dcfc189aeb07645739f8
ILP	protocol/contracts/interfaces/ILendingPool.sol	21affdbf3257e0e4f5f44963bdce91c88cfc48cfa386970f4a4ef0caec91dea6
ILA	protocol/contracts/interfaces/ILendingPoolAddressesProvider.sol	de02078a3fb2350b97a86b89d1de5ac0456de2d1c24c4d126245820cd8eba047
ILR	protocol/contracts/interfaces/ILendingPoolAddressesProviderRegistry.sol	0032d259d15c7f0c68687232347248c38ed856287ecf8d92d315cedef2ad9c1b
ILC	protocol/contracts/interfaces/ILendingPoolCollateralManager.sol	782eb7b1599a4bb5143ee1646a9675138b837226cc89bea44eabe6460a4f7f76
ILS	protocol/contracts/interfaces/ILendingPoolConfigurator.sol	3f7b6f5cd715ee471ae372c900ac07487d0a25d8e3cbc037c5330f1d88e26c28
ILO	protocol/contracts/interfaces/ILendingRateOracle.sol	2850412840ef98eb05e0920166575796775954768162be42461e4844ace73abf
IPA	protocol/contracts/interfaces/IPriceAggregatorAdapter.sol	097c7c4f40e90ff7d3852c86022586d12aca80ded8cba07f770733fbad9e940e

ID	File	SHA256 Checksum
IPO	protocol/contracts/interfaces/IPriceOracle.sol	ac59d2695dbf1683d5df2f4ec52f8b697efb55b4526f23ea35060978de68564f
IPG	protocol/contracts/interfaces/IPriceOracleGetter.sol	671ce54540c2e61cf04ce0e51079f8a2b550576a7dffc115b012aedb1840d5e3
IRI	protocol/contracts/interfaces/IReserveInterestRateStrategy.sol	dbbe1b34f677cd7e76d5a905d4659ee08943073b628840c847e924ed8a9bcfd3
ISB	protocol/contracts/interfaces/IScaledBalanceToken.sol	2c669c0b02e60f3fa3d3190b96a974652c0426b226c6f7a1870d10b16475288b
ISD	protocol/contracts/interfaces/IStableDebtToken.sol	7d4e3f3f1a38f1a293e501392539e2f856172e0d173d42741095cd065dbbf739
IST	protocol/contracts/interfaces/IStakedToken.sol	ac991c0903d85a39630a64a3269c02e7591d9a7fd2308c43309fdbca98fc1121
ISI	protocol/contracts/interfaces/IStarlayIncentivesController.sol	dc5247bbab3cca16e04cbbce0ec684e96b4047b5b30b16e8823ec0f2b571beca
IVD	protocol/contracts/interfaces/IVariableDebtToken.sol	9c362de55e164d9413d6c135268c41bfd4744c9d6d44819e601df0d26623b44e
SUI	protocol/contracts/interfaces/StakeUIHelperI.sol	bf0ae3a6fa65cf45220959d21f7fc8b1222532c1b80a42e8cc6fe472c59a2948
ISO	protocol/contracts/misc/interfaces/IStarlayOracle.sol	ed35083a9586ebe5e352d1a0c960499de824e3bcd5031010d02fb6e2c0241669
IUI	protocol/contracts/misc/interfaces/IUiIncentiveDataProviderV2.sol	a99e00dd950898fded2882ec4709b46aca7a76ee14f54f949df6dba2c54fb40c
IUP	protocol/contracts/misc/interfaces/IUiPoolDataProviderV2.sol	892a255aba543a514af4c1c9fd3c519e6314bb2690e1f3ecbcb8df6b68e6d2b4
IWE	protocol/contracts/misc/interfaces/IWETH.sol	17fad436e40decc68929dbd0dc8f18425c6dc33e4f4d12aeb12615e3586d7b28
IWT	protocol/contracts/misc/interfaces/IWETHGateway.sol	91318390009707b91207f3cb579bac9d5b3d5746ed6a23d941e8d114fcdfeb7a
PAA	protocol/contracts/misc/PriceAggregatorAdapterChainlinkImpl.sol	f9290f8918d97725a4e3adf2c418dcc12ed1453d45f5a75100718f618fb36792

ID	File	SHA256 Checksum
PAD	protocol/contracts/misc/PriceAggregatorAdapterDiaImpl.sol	05f0d5bea359f32aa39d2aea24f464bf8280b5b512089488e1f69cef6702b643
SUH	protocol/contracts/misc/StakeUIHelper.sol	863f0d57b537b2e717f15ba0c4428e1fe6af6344e06cda8cfec597cbc77229bd
SFO	protocol/contracts/misc/StarlayFallbackOracle.sol	f028024a015c7373919c1e077e35ebe8f70aa6f6d5bbef2a181599efe6ad40d9
SOS	protocol/contracts/misc/StarlayOracle.sol	716e51db3c9ed11154fc756a67ef0382f50a92fe643e3583498d8f929552bf41
SPD	protocol/contracts/misc/StarlayProtocolDataProvider.sol	5c1d5003904b471681dc8f5ef4b5272456de85803d7738f05a1773f9d647894d
UID	protocol/contracts/misc/UiIncentiveDataProviderV2.sol	0511d54f773aa89b304a4b1b7d9221af2038a641084ffad30c8c22f249c79b74
UPD	protocol/contracts/misc/UiPoolDataProviderV2.sol	4dea2f3215515f48241ee6519e8d543c4f71562b0f28a6085a9e83089d1c8a9b
WET	protocol/contracts/misc/WETHGateway.sol	94e326b149c84ef5466d6833a1973cfd9ae23b12f1ad248ae795bc54d0654f30
WBP	protocol/contracts/misc/WalletBalanceProvider.sol	b53ef6db62ce12b27710cbe6ee10602612c52a54081435d9fd6d476ab8a2af90
LPA	protocol/contracts/protocol/configuration/LendingPoolAddressesProvider.sol	d3d451e31e4a024b15e9599f88666664ae6de2d0f8a3753a49fca4160e033627
LPP	protocol/contracts/protocol/configuration/LendingPoolAddressesProviderRegistry.sol	bfd1fe86aa8c08993d4d31024ee5ed03246cad78e96ca8036544c09437aacdfa
DRI	protocol/contracts/protocol/lendingpool/DefaultReserveInterestRateStrategy.sol	f9a6878ce4798c01b1bb64f22a98d126d2dfedb2d8a90d70ec500a8e8d3ccaca
LPS	protocol/contracts/protocol/lendingpool/LendingPool.sol	927d9d63fb2f23235576b2afeaea71578fb89938b5ec181c20ec055cb08d61fd
LPC	protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol	5f1173b20ff8edf88fc1b212af14dd1ef31f340cc4068f681743e763e382a30d
LPK	protocol/contracts/protocol/lendingpool/LendingPoolConfigurator.sol	7e9ea4acdbfb7a2bc6b90a55d7914359ea9d0e661f2cf07e5c4c516817d1022f

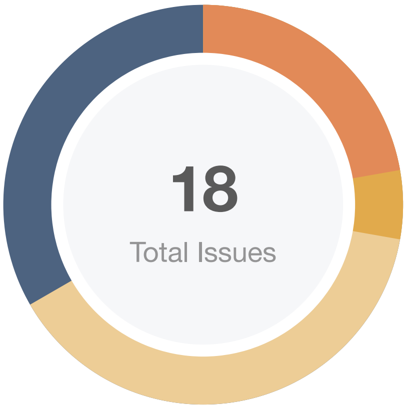
ID	File	SHA256 Checksum
LSS	protocol/contracts/protocol/lendingpool/LendingPoolStorage.sol	0ac9aa8ec5dd403526360ddc0670b8f7227f757b71ea42a4394575a0776a543e
RCS	protocol/contracts/protocol/libraries/configuration/ReserveConfiguration.sol	89046195aea0062f4aab5c74b137e506d4df6eed1d44f9582e39deb6609d52fa
UCS	protocol/contracts/protocol/libraries/configuration/UserConfiguration.sol	0e1138f8ced7be6718b26edfb84a4bbe17f97c5aabdfb97e510497bd4b36ec91
ESC	protocol/contracts/protocol/libraries/helpers/Errors.sol	df5acbb0d48aa57a0ab9d83f0183810ceda878fea6977b6c5fb0182650fc8e04
HSC	protocol/contracts/protocol/libraries/helpers/Helpers.sol	fb525a2f9648a9d5d7d2f12eabfd2b12d4b3f7a893fdcc7bbae3703e8bd5e366
GLS	protocol/contracts/protocol/libraries/logic/GenericLogic.sol	04de8c81851efc97ee4638d011ee2288677e50f6333645e9f62434bd3d6c70e0
RLS	protocol/contracts/protocol/libraries/logic/ReserveLogic.sol	5deba73c69c257b36cc8e46d1c5986ad1b793950f0a92cdfbf800f3ca09b8021
VLS	protocol/contracts/protocol/libraries/logic/ValidationLogic.sol	c895e136819db1250168d7f597bc58c7c8cf07b33211d7ec7ea73962819ff9fc
MUS	protocol/contracts/protocol/libraries/math/MathUtils.sol	d978e1b44e9a248749457bd2d6daecd95c70040c3cfd26679ad13e0cb64d5ea9
PMS	protocol/contracts/protocol/libraries/math/PercentageMath.sol	7605c89ecf0184f9e5bfe4bf8ae2c7e02f3631c848afe5acb8c99abb093e9a8a
WRM	protocol/contracts/protocol/libraries/math/WadRayMath.sol	257337f971837c06eb245533d9b0a4a35b65f8b6bb67aed7118b6e93b2cbdbfa
BIA	protocol/contracts/protocol/libraries/starlay-upgradeability/BaseImmutableAdminUpgradeabilityProxy.sol	fa2c9c7c24f49e9ba22c4580e076a076916e50742a299310511ba221ec13be14
IIA	protocol/contracts/protocol/libraries/starlay-upgradeability/InitializableImmutableAdminUpgradeabilityProxy.sol	cd47c181ab472cb601a32a864a4cffd0ea33b40cec30d4c17b2035ce91848b0e
VIC	protocol/contracts/protocol/libraries/starlay-upgradeability/VersionedInitializable.sol	4cdd06d87a4db4f6838fc93616fab3e0d38cc4fb01ce684f9213ef5317ed7b05
DTS	protocol/contracts/protocol/libraries/types/DataTypes.sol	6ce111ff49e846ffe6060b21d2d582343651b032b3b6c69db92982a65e24dc8a

ID	File	SHA256 Checksum
DTB	protocol/contracts/protocol/tokenization/base/DebtTokenBase.sol	33d085de31780ebe60906cecfa1f34532b279fcd5462ae9aee876db5415c39e0
DAL	protocol/contracts/protocol/tokenization/DelegationAwareLToken.sol	c60211e7fce9b9c5fa226fab3e372f9420f1e2d4d82326fd7fa3c6de4085d0cd
IRC	protocol/contracts/protocol/tokenization/IncentivizedERC20.sol	36436397289135d677632fe8e106e147f75618d69431aeba730aff1677fdc1bc
LTS	protocol/contracts/protocol/tokenization/LToken.sol	24e85be2578d77fcabc9dacfbd4cea2dbadc5568f176579683990cf892b4bcef
SDT	protocol/contracts/protocol/tokenization/StableDebtToken.sol	a0e2211d482b293513676f808a5d4501c0e6cb9f34e4734e571d0dfabb6a3e45
VDT	protocol/contracts/protocol/tokenization/VariableDebtToken.sol	d67864c2964eea463d1c38aa2ac8e0b12698c10a043ac91a16d758b94c62b714
IAT	stake/contracts/interfaces/IAToken.sol	6edace2fece605fcf6c946079803245560fabc9bd574f6343eabdaf86c20db84
IBP	stake/contracts/interfaces/IBPool.sol	08ebb8d15d9ef5ae3b063079693b5827270cf1ad53540dcd7faf37716538fb6a
ICR	stake/contracts/interfaces/ICRPFactory.sol	35b9c99ee52ec9393313e3df626bfa0f62ef97b39920ca8654dc048c95109ac2
ICP	stake/contracts/interfaces/IConfigurableRightsPool.sol	2e3f5b7c31266e83a6ad7eba33343c56060a32d8f31c610eeedd8eb6a05a2d84
ICS	stake/contracts/interfaces/IControllerStarlayEcosystemReserve.sol	c77f4acc31056a2a0da66362fe02bb0c43267e96c3a8ac0a94073c22e03e76cc
IDA	stake/contracts/interfaces/IDelegationAwareToken.sol	50b6017c43921266b436bb230b9eaea24ed7d48b8efab057c0918501771602c2
IDM	stake/contracts/interfaces/IDistributionManager.sol	85e65acf304a0e2e3bafdf480ca52ea0e36954db821a44b45b0b0c58d4c538dc
IER	stake/contracts/interfaces/IERC20.sol	42c346056d38377aa31f6ff514b03c5afdf3ea0a6c71390f7fbfdde8e36d4b0d
IEC	stake/contracts/interfaces/IERC20Detailed.sol	0e62e6c1c1fc76fefe561d61a33cdaabe65b55aab5d7af8dc2aa385ecae12079

ID	File	SHA256 Checksum
IIC	stake/contracts/interfaces/IIncentivesController.sol	b844156d7f98c10d7ca15636a30e3cdde7175417db29180d7f06afc14d66d386
ISL	stake/contracts/interfaces/IStakedLay.sol	643b114a6a0ffa46727d2807f54d4a5153f9d8c2b73e3f629bce1a5720a719bd
ISG	stake/contracts/interfaces/IStarlayGovernanceV2.sol	b910796951ff878d2849efaf9a90f7fba408cda67dce3c73fb6eb2fde7adac1e
ITH	stake/contracts/interfaces/ITransferHook.sol	bdf4458a0f6ac13cd43ae250e7d07da5e019e12dc1f2df0afce1e35e8769b415
IES	stake/contracts/stake-v1/contracts/interfaces/IERC20.sol	e3ca5f624276b31f09be8627baade41f7a94c3f7cde4dcf8cab684e50e2b9817
IGP	stake/contracts/stake-v1/contracts/interfaces/IGovernancePowerDelegationToken.sol	e45353c4170a412910d2c138195986400c1be3a80d10d602d1eb50113062f08d
ASC	stake/contracts/stake-v1/contracts/open-zeppelin/Address.sol	07df4785d73cdd956f68f75623e2e841ed272c908946afd9ee4636ab27c633a7
CSC	stake/contracts/stake-v1/contracts/open-zeppelin/Context.sol	4721ba0a55289e6806974759a570e3eeff6b8b6660c15d40e6f16992bd35dea1
ERC	stake/contracts/stake-v1/contracts/open-zeppelin/ERC20.sol	292b623b04572da22c54465860c0a7e4c6ba8f43b1b39011bc06089921a7ca1b
SMS	stake/contracts/stake-v1/contracts/open-zeppelin/SafeMath.sol	b0380ce065d45c348efc5cbb85c510f758f5c6246a0086c9b8faace31d70b2e3
GPD	stake/contracts/stake-v1/contracts/token/base/GovernancePowerDelegationERC20.sol	50b453744c3ba891c04ba4f46782cfcf44d8f17a6f0a1e7468b1809f41029dfd
DTH	stake/contracts/stake-v1/contracts/utils/DoubleTransferHelper.sol	eb5a5ae311bd656d4da01f4e53683cf92c7d31763c238855b1c650cb368e0091
DMS	stake/contracts/stake/DistributionManager.sol	1029631929190768df80b1c27b81562e1c692d772513593c57de89fe57e10c4d
ICC	stake/contracts/stake/IncentivesController.sol	52f497f99004a3046b1467f48b918fce4d69295cf129757adc52512db93ab4a4
SLS	stake/contracts/stake/StakedLay.sol	98254f9d46cf0efa6e0726224480ee2c146606cbe0e75880cb7dce5aa0664de3

ID	File	SHA256 Checksum
SLV	stake/contracts/stake/StakedLayV2.sol	a04fc4e8d760e4d00035ba1a91d3b461a8de7c54a0a486f75a7200ad6248dc18
STS	stake/contracts/stake/StakedToken.sol	99a264d697110d70c5f73926c64f9e8e90a082ef8f482d6b43023128356cb3d8
STV	stake/contracts/stake/StakedTokenV2.sol	1e344a99089d7991b47eadd511750875818da0f8c08cc1526195768bcb7c8c41
STC	stake/contracts/stake/StakedTokenV3.sol	d1ba9294027c5655d411014447e34344c13f3084bc06790d6211c2f234b3011e
MES	stake/contracts/utils/MintableErc20.sol	b69f94e045244931adb74483631011e99c1e0f4d8ebb527296d08fc6ac9b9bd0
VIS	stake/contracts/utils/VersionedInitializable.sol	f7cd8f414fb49e10e52ad37ee89d31c031c54144d6b644224a7100ab8df1e118
SRV	stake/contracts/vault/StarlayRewardsVault.sol	bb46ca77d4d4d5e4742cad83e5fa1b1d5bbd2d59e151a6a1ab96c4b87ef69eca
TSC	stake/contracts/vesting/Token.sol	2cea195041092946989fae3650ab1480f771bfff7c5a8027756a9ad4efbaca75
TVS	stake/contracts/vesting/TokenVesting.sol	9ed0e5354688e2b5be2fd2a7c3fa3ebbc4bfecdf7446bb73ca10c8730c40592e

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**4** (22.22%)
🟨 **Medium**	**1** (5.56%)
🟨 **Minor**	**7** (38.89%)
🟦 **Informational**	**6** (33.33%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GPD-01	Discussion about Function `totalSupplyAt()`	Logical Issue	● Informational	ⓘ Acknowledged
INC-01	Centralization Related Risks	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
LPS-01	Potential Flashloan Attack	Logical Issue	● Medium	ⓘ Acknowledged
LPS-02	Incompatibility With Deflationary Tokens	Logical Issue	● Minor	ⓘ Acknowledged
MEC-01	Unused Parameter	Gas Optimization	● Informational	✓ Resolved
PAA-01	Third Party Dependencies	Volatile Code	● Minor	ⓘ Acknowledged
SCK-01	Centralization Related Risks	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
SCP-01	Centralization Related Risks	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
SKP-01	Lack of Zero Address Validation	Volatile Code	● Minor	✓ Resolved
SKP-02	Potential Front-Running Risk	Volatile Code	● Minor	ⓘ Acknowledged
SKP-03	Incompatibility With Deflationary Tokens	Logical Issue	● Minor	ⓘ Acknowledged
SKP-04	Possibility of Replay Attack in `Permit`	Volatile Code	● Minor	✓ Resolved
SKP-05	Susceptible to Signature Malleability	Volatile Code	● Minor	✓ Resolved

ID	Title	Category	Severity	Status
SKP-06	Missing Emit Events	Coding Style	● Informational	✓ Resolved
SKP-07	Lack of Access Control	Logical Issue	● Informational	✓ Resolved
TSC-01	Initial Token Distribution	**Centralization / Privilege**	● **Major**	✓ Resolved
TVS-01	Lack of Error Message	Coding Style	● Informational	✓ Resolved
TVS-02	Comparison to A Boolean Constant	Gas Optimization	● Informational	✓ Resolved

GPD-01 | Discussion About Function `totalSupplyAt()`

Category	Severity	Location	Status
Logical Issue	● Informational	stake/contracts/stake-v1/contracts/token/base/GovernancePowerDelegationERC20.sol: 108	ⓘ Acknowledged

Description

The function `totalSupplyAt()` returns the total supply at a certain block number, however, it actually returns the total supply.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

`[Client]`: As mentioned in the recommendation, the original project has the function as it is. We suspect this function is called by a third party because OpenZeppelin prepares the interfaces:ERC 20 - OpenZeppelin Docs
It is correct that it does not work as intended with an unnecessary parameter, but as long as the total supply may be called using this interface, we would say it is better to leave it as is.

INC-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	● Major	incentives-controller/contracts/incentives/base/DistributionManager.sol: 38 incentives-controller/contracts/incentives/base/BaseIncentivesController.sol: 45, 108, 129 incentives-controller/contracts/incentives/PullRewardsIncentivesController.sol: 50	ⓘ Acknowledged

Description

The `EMISSION_MANAGER` of the contract `StakedTokenIncentivesController` has the responsibility to notify users about the following capabilities:

- set `_distributionEnd` through `setDistributionEnd()`
- set claimer through `setClaimer()`
- configures the distribution of rewards for a list of assets through `configureAssets()`

The `EMISSION_MANAGER` of the contract `PullRewardsIncentivesController` has the responsibility to notify users about the following capabilities:

- set `_distributionEnd` through `setDistributionEnd()`
- set claimer through `setClaimer()`
- configures the distribution of rewards for a list of assets through `configureAssets()`
- set `_rewardsVault` through `setRewardsVault()`

The `_authorizedClaimers` of the contract `StakedTokenIncentivesController`/`PullRewardsIncentivesController` has the responsibility to notify users about the following capabilities:

- claim rewards on behalf through `claimRewardsOnBehalf()`

Any compromise to the `EMISSION_MANAGER`/`_authorizedClaimers` account may allow a hacker to take advantage of this authority.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles. OR
- Remove the risky functionality.

Alleviation

`[Client]`: Currently, on Astar, multi-sig wallets such as GnosisSafe are not officially supported, and secure contract wallets are not available. In that situation, we avoid having our keys stolen by using Ledger hardware wallets.

LPS-01 | Potential Flashloan Attack

Category	Severity	Location	Status
Logical Issue	● Medium	protocol/contracts/protocol/lendingpool/LendingPool.sol: 299	ⓘ Acknowledged

Description

The function can be affected by a flashloan attack such that the new stable borrow rate becomes the lowest possible. A malicious actor can first request a flashloan to deposit into the reserve pool so that the reserve's utilization rate is close to 0, then invoke `swapBorrowRateMode()` to perform the variable-to-borrow rate switch and enjoy the lowest `currentStableBorrowRate`, and finally withdraw to return the flashloan. A similar approach can also be applied to bypass `maxStableLoanPercent` enforcement in `validateBorrow()`.

Recommendation

We advise the client to revise logic to defensively detect sudden changes to a reserve utilization and block malicious attempts.

Alleviation

No alleviation.

LPS-02 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	protocol/contracts/protocol/lendingpool/LendingPool.sol: 106	ⓘ Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, an inconsistency in the amount will occur and the transaction may fail due to the validation checks.

Recommendation

We advise the client to regulate tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

`[Client]`: We do not plan to support deflationary tokens but understand the need to keep this in mind as tokens are added.

MEC-01 | Unused Parameter

Category	Severity	Location	Status
Gas Optimization	● Informational	incentives-controller/contracts/utils/MintableErc20.sol: 26	✓ Resolved

Description

The parameter `deadline`, `v`,`r` and `s` are not used in the function.

Recommendation

We advise the client to remove them if they are not intended to be used.

Alleviation

The client revised the code and resolved this issue.

PAA-01 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	protocol/contracts/misc/PriceAggregatorAdapterChainlinkImpl.sol: 18	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party protocols such as Price Oracle, including:

- `IChainlinkAggregator.latestAnswer()`

Recommendation

We understand that the business logic requires interaction with the aforementioned protocols. We encourage the team to constantly monitor the status of 3rd parties to mitigate side effects when unexpected activities are observed.

Alleviation

No alleviation.

SCK-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	stake/contracts/vesting/TokenVesting.sol: 139, 179, 199 stake/contracts/vault/StarlayRewardsVault.sol: 9, 13 stake/contracts/stake/DistributionManager.sol: 45	ⓘ Acknowledged

Description

The `owner` of the contract `TokenVesting` has the responsibility to notify users about the following capabilities:

- creates a new vesting schedule for a beneficiary through `createVestingSchedule()`
- revokes the vesting schedule for the given identifier through `revoke()`
- withdraw the specified amount to himself/herself through `withdraw()`

The `owner` of the contract `StarlayRewardsVault` has the responsibility to notify users about the following capabilities:

- set `incentiveController` through `setIncentiveController()`
- transfer uncapped tokens to `incentiveController` through `transfer()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority.

The `EMISSION_MANAGER` of the contract `DistributionManager` has the responsibility to notify users about the following capabilities:

- configures the distribution of rewards for a list of assets through `configureAssets()`

Any compromise to the `EMISSION_MANAGER` account may allow a hacker to take advantage of this authority.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles. OR
- Remove the risky functionality.

Alleviation

`[Client]`: Currently, on Astar, multi-sig wallets such as GnosisSafe are not officially supported, and secure contract wallets are not available. In that situation, we avoid having our keys stolen by using Ledger hardware wallets.

SCP-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	● Major	protocol/contracts/protocol/configuration/LendingPoolAddressesProviderRegistry.sol: 47, 59 protocol/contracts/protocol/configuration/LendingPoolAddressesProvider.sol: 47, 60, 75, 101, 119, 153, 162, 171, 180 protocol/contracts/protocol/tokenization/LToken.sol: 122, 146, 169, 193, 315, 330 protocol/contracts/protocol/tokenization/DelegationAwareLToken.sol: 27 protocol/contracts/protocol/tokenization/VariableDebtToken.sol: 95, 124 protocol/contracts/protocol/tokenization/StableDebtToken.sol: 136, 197 protocol/contracts/protocol/lendingpool/LendingPoolConfigurator.sol: 65, 149, 177, 212, 249, 267, 285, 333, 347, 361, 375, 392, 406, 421, 438, 438, 448 protocol/contracts/protocol/lendingpool/LendingPool.sol: 786, 809, 823, 836 protocol/contracts/misc/WETHGateway.sol: 30, 151, 165 protocol/contracts/misc/StarlayOracle.sol: 47, 54 protocol/contracts/misc/StarlayFallbackOracle.sol: 33, 39, 45 protocol/contracts/deployments/StableAndVariableTokensHelper.sol: 21, 29, 42 protocol/contracts/misc/PriceAggregatorAdapterDiaImpl.sol: 32, 42 protocol/contracts/misc/PriceAggregatorAdapterChainlinkImpl.sol: 24 protocol/contracts/deployments/LTokensAndRatesHelper.sol: 48, 67 protocol/contracts/protocol/libraries/starlay-upgradeability/BaseImmutableAdminUpgradeabilityProxy.sol: 34, 41, 50, 63	ⓘ Acknowledged

Description

The `owner` of the contract `LendingPoolAddressesProviderRegistry` has the responsibility to notify users about the following capabilities:

- registers an addresses provider through `registerAddressesProvider()`
- removes a LendingPoolAddressesProvider from the list of registered addresses provider through `unregisterAddressesProvider()`

The `owner` of the contract `LendingPoolAddressesProvider` has the responsibility to notify users about the following capabilities:

- allows to set the market which this LendingPoolAddressesProvider represents through `setMarketId()`
- updates the implementation of a proxy registered with certain `id` through `setAddressAsProxy()`
- sets an address for an id replacing the address saved in the addresses map through `setAddress()`
- updates the implementation of the LendingPool, or creates the proxy through `setLendingPoolImpl()`
- updates the implementation of the LendingPoolConfigurator, or creates the proxy through `setLendingPoolConfiguratorImpl()`
- updates the address of the LendingPoolCollateralManager through `setLendingPoolCollateralManager()`
- sets pool admin through `setPoolAdmin()`
- sets emergency admin through `setEmergencyAdmin()`
- sets price oracle through `setPriceOracle()`
- sets lending rate oracle through `setLendingRateOracle()`

The `owner` of the contract `WETHGateway` has the responsibility to notify users about the following capabilities:

- authorize lending pool through `authorizeLendingPool()`
- transfer ERC20 from the utility contract through `emergencyTokenTransfer()`
- transfer native Ether from the utility contract through `emergencyEtherTransfer()`

The `owner` of the contract `StarlayOracle` has the responsibility to notify users about the following capabilities:

- sets `_adapter` through `setPriceAggregator()`
- sets `_fallbackOracle` through `setFallbackOracle()`

The `owner` of the contract `StarlayFallbackOracle` has the responsibility to notify users about the following capabilities:

- authorizes `Sybil` through `authorizeSybil()`
- unauthorizes `Sybil` through `unauthorizeSybil()`

The `owner` of the contract `StableAndVariableTokensHelper` has the responsibility to notify users about the following capabilities:

- inits deployment through `initDeployment()`

- sets oracle borrow rates through `setOracleBorrowRates()`
- sets oracle ownership through `setOracleOwnership()`

The `owner` of the contract `PriceAggregatorAdapterDiaImpl` has the responsibility to notify users about the following capabilities:

- sets `_aggregator` through `setAggregator()`
- sets or replaces sources of assets through `setAssetSources()`

The `owner` of the contract `PriceAggregatorAdapterChainlinkImpl` has the responsibility to notify users about the following capabilities:

- sets or replaces sources of assets through `setAssetSources()`

The `owner` of the contract `LTokensAndRatesHelper` has the responsibility to notify users about the following capabilities:

- inits deployment through `initDeployment()`
- configures reserves through `configureReserves()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority.

The `lendingPool` of the contract `LToken`/`DelegationAwareLToken` has the responsibility to notify users about the following capabilities:

- burns `lTokens` from `user` and sends the equivalent amount of underlying to `receiverOfUnderlying` through `burn()`
- mints `amount` `lTokens` to `user` through `mint()`
- mints `lTokens` to the reserve treasury through `mintToTreasury()`
- transfers `lTokens` in the event of a borrow being liquidated, in case the liquidators reclaims the `lToken` through `transferOnLiquidation()`
- transfers the underlying asset to `target` through `transferUnderlyingTo()`
- invoked to execute actions on the `lToken` side after a repayment through `handleRepayment()`

The `lendingPool` of the contract `VariableDebtToken`/`StableDebtToken` has the responsibility to notify users about the following capabilities:

- mints debt token to the `onBehalfOf` address through `mint()`
- burns user variable debt through `burn()`

Any compromise to the `lendingPool` account may allow a hacker to take advantage of this authority.

The `poolAdmin` of the contract `DelegationAwareLToken` has the responsibility to notify users about the following capabilities:

- delegates voting power of the underlying asset to a `delegatee` address through `delegateUnderlyingTo()`

The `poolAdmin` of the contract `LendingPoolConfigurator` has the responsibility to notify users about the following capabilities:

- initializes reserves in batch through `batchInitReserve()`
- updates the `lToken` implementation for the reserve through `updateLToken()`
- updates the stable debt token implementation for the reserve through `updateStableDebtToken()`
- updates the variable debt token implementation for the asset through `updateVariableDebtToken()`
- enables borrowing on a reserve through `enableBorrowingOnReserve()`
- disables borrowing on a reserve through `disableBorrowingOnReserve()`
- configures the reserve collateralization parameters through `configureReserveAsCollateral()`
- enable stable rate borrowing on a reserve through `enableReserveStableRate()`
- disable stable rate borrowing on a reserve through `disableReserveStableRate()`
- activates a reserve through `activateReserve()`
- deactivates a reserve through `deactivateReserve()`
- freezes a reserve through `freezeReserve()`
- unfreezes a reserve through `unfreezeReserve()`
- updates the reserve factor of a reserve through `setReserveFactor()`
- sets the interest rate strategy of a reserve through `setReserveInterestRateStrategyAddress()`

Any compromise to the `poolAdmin` account may allow a hacker to take advantage of this authority.

The `emergencyAdmin` of the contract `LendingPoolConfigurator` has the responsibility to notify users about the following capabilities:

- pauses or unpauses all the actions of the protocol, including `lToken` transfers through `setPoolPause()`

Any compromise to the `emergencyAdmin` account may allow a hacker to take advantage of this authority.

The `lendingPoolConfigurator` of the contract `LendingPool` has the responsibility to notify users about the following capabilities:

- initializes a reserve, activating it, assigning a `lToken` and debt tokens and an interest rate strategy through `initReserve()`

- updates the address of the interest rate strategy contract through `setReserveInterestRateStrategyAddress()`
- sets the configuration bitmap of the reserve as a whole through `setConfiguration()`
- sets the _pause state of a reserve through `setPause()`

Any compromise to the `lendingPoolConfigurator` account may allow a hacker to take advantage of this authority.

The `Admin` of the contract `BaseImmutableAdminUpgradeabilityProxy` has the responsibility to notify users about the following capabilities:

- views the address of the proxy admin through `admin()`
- views the address of the implementation through `implementation()`
- upgrades the backing implementation of the proxy through `upgradeTo()`
- upgrades the backing implementation of the proxy and calls a function on the new implementation through `upgradeToAndCall()`

Any compromise to the `Admin` account may allow a hacker to take advantage of this authority.

The `Sybil` of the contract `StarlayFallbackOracle` has the responsibility to notify users about the following capabilities:

- submits prices through `submitPrices()`

Any compromise to the `Sybil` account may allow a hacker to take advantage of this authority.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles. OR
- Remove the risky functionality.

Alleviation

`[Client]`: Currently, on Astar, multi-sig wallets such as GnosisSafe are not officially supported, and secure contract wallets are not available. In that situation, we avoid having our keys stolen by using Ledger hardware wallets.

SKP-01 | Lack Of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	● Minor	stake/contracts/vesting/TokenVesting.sol: 139 stake/contracts/vault/StarlayRewardsVault.sol: 9 stake/contracts/stake/DistributionManager.sol: 36 stake/contracts/stake/StakedToken.sol: 52 stake/contracts/stake/IncentivesController.sol: 39 incentives-controller/contracts/incentives/PullRewardsIncentivesController.sol: 32 incentives-controller/contracts/incentives/StakedTokenIncentivesController.sol: 31 protocol/contracts/protocol/tokenization/LToken.sol: 66 protocol/contracts/misc/StakeUIHelper.sol: 21	✓ Resolved

Description

Addresses should be checked before assigning to make sure they are not zero addresses.

Recommendation

We advise the client to add validation to check them.

Alleviation

The client revised the code and resolved this issue.

SKP-02 | Potential Front-Running Risk

Category	Severity	Location	Status
Volatile Code	● Minor	stake/contracts/stake/StakedToken.sol: 78 stake/contracts/stake/StakedTokenV2.sol: 101 stake/contracts/stake/StakedTokenV3.sol: 101 stake/contracts/stake/IncentivesController.sol: 56 incentives-controller/contracts/incentives/PullRewardsIncentivesController.sol: 32 incentives-controller/contracts/incentives/StakedTokenIncentivesController.sol: 31 protocol/contracts/protocol/tokenization/LToken.sol: 66 protocol/contracts/protocol/tokenization/StableDebtToken.sol: 41 protocol/contracts/protocol/tokenization/VariableDebtToken.sol: 35 protocol/contracts/protocol/lendingpool/LendingPoolConfigurator.sol: 57 protocol/contracts/protocol/lendingpool/LendingPool.sol: 88	ⓘ Acknowledged

Description

Malicious hackers may observe the pending transaction which will execute the `initialize` function and launch a similar transaction with the hacker's address, set variables of the contract.

Recommendation

We advise the client to design functionality to only allow a specific user to execute the `initialize` function.

Alleviation

`[Client]`: These three responses are no longer necessary: StakedToken, StakedTokenV2, and StakedTokenV3. We think it is no issue that others can call it in the `IncentivesController.sol`. Other initializers have already initialized.

SKP-03 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	stake/contracts/stake/StakedToken.sol: 90 stake/contracts/stake/StakedTokenV2.sol: 120 stake/contracts/stake/StakedTokenV3.sol: 129	ⓘ Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, an inconsistency in the amount will occur and the transaction may fail due to the validation checks.

Recommendation

We advise the client to regulate tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

`[Client]`: We do not plan to support deflationary tokens but understand the need to keep this in mind as tokens are added.

SKP-04 | Possibility Of Replay Attack In `Permit`

Category	Severity	Location	Status
Volatile Code	● Minor	stake/contracts/stake/StakedTokenV2.sol: 347 stake/contracts/stake/StakedTokenV3.sol: 357 protocol/contracts/protocol/tokenization/LToken.sol: 343	✓ Resolved

Description

The `permit` function performs the operation of deriving signer address from the signature values of `v`, `r` and `s`. The state variable `DOMAIN_SEPARATOR` that is used to calculate hash has a value of `chainid` that is derived only once in the constructor, which does not change after contract deployment. The issue arises in the event of fork when the cross-chain replay attacks can be executed. The attack scenario can be thought of as if a fork of Ethereum happens and two different networks have id of for example `1` and `9`. The `chainid` coded in `DOMAIN_SEPARATOR` will be the same on contracts residing in both of the forks. If the chainid `1` is stored in the contract then the `permit` transaction signed for chainid `1` will be executable on both of the forks.

Recommendation

We advise to construct the `DOMAIN_SEPARATOR` hash inside the permit function so the current `chainid` could be fetched and only the transactions signed for current network could succeed.

Alleviation

The client revised the code and resolved this issue.

SKP-05 | Susceptible To Signature Malleability

Category	Severity	Location	Status
Volatile Code	● Minor	stake/contracts/stake/StakedTokenV2.sol: 460, 490 stake/contracts/stake/StakedTokenV3.sol: 470, 500	✓ Resolved

Description

The signature malleability is possible within the Elliptic Curve cryptographic system. An Elliptic Curve is symmetric on the X-axis, meaning two points can exist with the same `X` value. In the `r`, `s` and `v` representation this permits us to carefully adjust `s` to produce a second valid signature for the same `r`, thus breaking the assumption that a signature cannot be replayed in what is known as a replay-attack.

Recommendation

We advise to utilize a `recover()` function similar to that of the `ECDSA.sol` implementation of OpenZeppelin.

Alleviation

The client revised the code and resolved this issue.

SKP-06 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	stake/contracts/vault/StarlayRewardsVault.sol: 9 stake/contracts/vesting/TokenVesting.sol: 139, 179, 199 protocol/contracts/misc/PriceAggregatorAdapterDiaImpl.sol: 25, 32 protocol/contracts/misc/PriceAggregatorAdapterChainlinkImpl.sol: 24	✓ Resolved

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers.

Recommendation

We advise the client to add events for sensitive actions and emit them.

Alleviation

The client revised the code and resolved this issue.

SKP-07 | Lack Of Access Control

Category	Severity	Location	Status
Logical Issue	● Informational	stake/contracts/utils/MintableErc20.sol: 22 stake/contracts/stake-v1/contracts/utils/DoubleTransferHelper.sol: 14 incentives-controller/contracts/utils/MintableErc20.sol: 18	✓ Resolved

Description

1. The function `mint()` can be called by anyone to mint tokens for themselves.
2. The function `doubleSend()` can be called by anyone to transfer the contract's tokens to anyone.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

[Client]: Deleted the contract `DoubleTransferHelper.sol` because it is not used. `MintableErc20.sol` is just for tests, and we remain it as it is.

TSC-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	● Major	stake/contracts/vesting/Token.sol: 13	✓ Resolved

Description

`initialSupply` tokens were sent to the `msg.sender` when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

The client revised the code and resolved this issue.

TVS-01 | Lack Of Error Message

Category	Severity	Location	Status
Coding Style	● Informational	stake/contracts/vesting/TokenVesting.sol: 55, 63, 64, 73	✓ Resolved

Description

`require` can be used to check for conditions and throw an exception if the condition is not met, in which case the descriptive error message provided by the developer will appear and help to track error and debugging.

Recommendation

We advise the client to add error messages.

Alleviation

The client revised the code and resolved this issue.

TVS-02 | Comparison To A Boolean Constant

Category	Severity	Location	Status
Gas Optimization	● Informational	stake/contracts/vesting/TokenVesting.sol: 55, 63, 64, 185	✓ Resolved

Description

Comparison to a boolean constant.

Recommendation

We advise the client to remove the comparison to the boolean constant.

Alleviation

The client revised the code and resolved this issue.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.